NuGenDEMVis

The aim of the project is to take 2D data sources and produce displacement mapped geometry and render it in a 3D context, whilst providing view and manipulation options.

# 1.0 Target Usage

1. Visualizing any sort of scientific data, just need to define a profile for each type
2. Visualizing GIS data, this might include small scale as well as large scale data. Mapping onto a sphere is totally possible (and cool), as long as we have coordinates, this fits in with having multiple overlapping data-sources of different resolution/detail.
3. Visualizing medical data, such as x-rays, scans etc. Although any 2D data representing a volume isn’t going to visualize very accurately/well.

# 2.0 Features

This is a rough list, which I want to try and stick with once we agree on it and start work. Additional items can of course be added once the core features are done and we have a reasonably stable version going.

## 2.1 Core Features

Things that are quite essential to the app.

1. Reading a variety of image formats
2. Reading a variety of data formats (GIS etc.)
3. Transforming the 2D data into a 3D representation
   1. Sampling options
   2. Control over transformation
   3. Support for transformation profiles
4. Generation of diffuse-maps from the 2D data
5. Some simple lighting and shading options
6. Support to take cross-sections of the 3D data (and 2D data)
7. MDI based UI
8. Various small tools for manipulating the geometry

## Extended Features

Things that are nice to have, but not that essential at the moment.

1. GPU based rendering of the 3D data (i.e. clipmaps)
2. Some sort of ray-tracing support, for large resolution static renders
3. Enhancement of geometry shading through normal-maps
4. Enhancement of geometry lighting through shadow-maps
5. Support for large scenes, possibly composed of multiple data sources
6. Exporting of data into various formats
7. Some sort of WPF support
8. Scripting support
9. Dynamic LOD for default rendering geometry

# Current Project Status

This is a bit rough, as I’ve forgotten exactly where we are. All in all I think we’re not doing too bad, if we can focus our efforts on the core items that really count then we’re not a million miles away from something usable. Coming back to the project fresh and with new ideas I think will really help get it moving I hope.

1. *Reading a variety of image formats.*  
    This is pretty good, all done via [GDAL](http://www.gdal.org/) so it already supports pretty much everything we want, both image formats and data formats.
2. *Reading a variety of data formats (GIS etc.).*Same as above.
3. *Transforming the 2D data into a 3D representation.*As I recall we have it doing this pretty well, with a couple of basic profiles, along with multi-sampling support.
   1. *Sampling options.*Not much currently exposed to the UI, only the sampling type (grey-scale etc.)
   2. *Control over transformation.*Nothing at the moment.
   3. *Support for transformation profiles.*Some basic profiles are supported I think, but we need to make this more comprehensive, along with an editor.
4. *Generation of diffuse-maps from the 2D data.*  
   As I recall this is not too bad, we have support for the original RGB if it’s an image, height-bands, 1-colour height. We’ll need to expand this a bit and improve things as well.
5. *Some simple lighting and shading options.*  
   Quite easy to add support for this, it’s all but there is just in need of a UI.
6. *Support to take cross-sections of the 3D data (and 2D data).*  
   I’d started on this a bit, but I need to finalize the geometry creation a bit more first before I can take it any further. As it is the 2D rendering of the cross-section is not looking too bad, complete with b-spline smoothing, but this needs to be improved along with implementation of the section capture etc.
7. *MDI based UI.*  
   Not looking too bad, it’s all essentially there. Things need to be improved, several parts are not currently functional such as visualization previews etc.
8. *Various small tools for manipulating the geometry.*  
   This includes camera and scene manipulation, along with height and distance measuring markers. Some parts are near implemented as I recall, but things need improving and adding here.
9. *GPU based rendering of the 3D data (i.e. clipmaps).*  
   I’d started this recently as you know, both a geometry-clipmap prototype and a simpler tree-culling approach. As it turned out both we’re quite complex and not too easy. As it stands I think we’ll have to go with the geo-clips for the GPU based rendering, I need to get to grips with the code again and work out changes needed to fit our data input. The tree-based approach I think will best be suited to the default rendering technique, for a LOD implementation.
10. *Some sort of ray-tracing support, for large resolution static renders.*This code is pretty much all there, I have the RT code and the code to render a height-map from the prototype/RT test app; it just needs integrating with our framework.
11. *Enhancement of geometry shading through normal-maps.*  
    This is pretty easy to do; I’d started it previously but was getting tripped up by the complex scene manager when it came to generating the normal-maps, I think I’ll have to refactor things to make it easier to do.
12. *Enhancement of geometry lighting through shadow-maps.*  
    I think out best bet here is perspective shadow-mapping, I’ll have to research it further though.
13. *Support for large scenes, possibly composed of multiple data sources.*  
    The data spatial manager for this is already there and in use, I think I need to take another look at it and improve existing things and finish off a few others, to give us something robust to work from.
14. *Exporting of data into various formats.*  
    Totally possible, not even looked at implementing it yet however, GDAL will probably help us a lot here.
15. *Some sort of WPF support.*  
    We have code to bridge WPF with (in-theory) our framework, as I recall it just needs integrating.
16. *Scripting Support.*  
    This is looking quite good, the editor, compiler etc. Are all there it’s just a case of defining what we want it to do and integrating support.
17. *Dynamic LOD for default rendering geometry.*  
    This is going to have to be CPU geometry-clipmaps or a tree clipping algorithm.

## Work Schedule / Time-plan

Trying to define the order of work, and eventually the rough time it should take for each section to be completed.

### Milestone 1 – All core features present and working roughly right

1. Tidy-up of the code
2. Expand profiles and new visualization options
3. Add more control over the generated geometry
4. Greatly improve diffuse-map generation and options etc.
5. Add & improve tools support
6. Add some lighting & shading options to UI
7. Formally add support for cross-sections
8. Get UI working robustly

### Milestone 2 – All Core features working well